

FAIRLY PARTITIONING RESOURCES WHILE LIMITING THE MAXIMUM FAIR SHARE

INVENTORS

Pawan Goyal

Srinivasan Keshav

Prepared by:

Renee M. DuBord

Reg. No. 42,500

Fenwick & West LLP

Two Palo Alto Square

Palo Alto, CA 94306

Express Mail No.: EL482474496US

FAIRLY PARTITIONING RESOURCES WHILE LIMITING THE MAXIMUM FAIR SHARE

INVENTORS

5 Pawan Goyal and Srinivasan Keshav

BACKGROUND

Field of Invention

10 The present invention relates generally to resource scheduling, and more particularly, to scheduling a resource fairly while preventing resource users from exceeding a maximum resource allotment.

Background of the Invention

15 A resource scheduler performs the function of allocating resources. Different resource types may use separate resource schedulers. Within each resource scheduler, each customer or user of the resource is treated as a separate "schedulable entity" with a separate scheduling queue and an individual quality of service guarantee. The scheduler selects requests for service from among the different queues, using a scheduling algorithm to ensure that each queue receives a least the minimum level of service they were guaranteed. Different queues may have different minimum quality of service guarantees, and the resource requests from each queue are weighted by the quality of service guarantee. Weighting increases or decreases a schedulable entity's
20 relative resource share.

The goal of the resource scheduler is twofold. First, the resource scheduler must try to ensure that each schedulable entity is allocated resources corresponding to at least the minimum quality of service resource level paid for by that schedulable entity. However, if extra resources are available, the resource scheduler must decide how to allocate the additional resources.

5 Typical resource scheduling algorithms and methods are work-conserving. A work-conserving scheduler is idle only when there is no resource request to service. Additional resources will be divided up among the schedulable entities with outstanding requests, in proportion to each schedulable entity's weight. Customer requests are serviced if the resource is available, even if they exceed the schedulable entities' quality of service guarantee.

Thus, a work-conserving resource scheduler may often provide schedulable entities with service beyond the actual maximum quality of service paid for by the schedulable entity. Unfortunately, this behavior tends to create unrealistic expectations. For example, assume customers A and B both are sharing a resource. Customer A pays for 50% of the resources and customer B pays for 25%, and resource sharing is weighted between A and B to reflect these different allocations. In a work-conserving scheduler, customers with unsatisfied requests get resource shares in proportion to their weights. Therefore, if both A and B request resources beyond their paid-for quality of service guarantee, one embodiment of a work-conserving scheduler will allocate two-thirds of the extra 25% of the resources to A, and the remaining one-third to B.

20 However, if a new customer C is added to further share the resource, and C pays for and receives 25% of the resources, the resources actually delivered to A and B will decrease. Although A and B will still receive the 50% and 25%, respectively, of resources that they paid for, A and B may both perceive a decrease in service. In order to avoid setting unrealistic

customer expectations, it is preferable for the resource scheduler to prevent customers from receiving more resources than they have paid for, even if this allows resources to be idle during certain times. Such a preferred resource scheduler is non-work-conserving, and implements both minimum and maximum quality of service guarantees.

5 One existing method for preventing schedulable entities from exceeding their maximum allotted resources is to place a rate controller module into the system before the resource scheduler. The rate controller module receives resource requests from each different schedulable entity and separates them into individual "schedulable entity" queues. Each schedulable entity has a separate queue. Before a request is passed on to the scheduler, the rate controller module checks to determine if granting the request will exceed the schedulable entity's maximum resource allotment. If so, the rate controller module sets a timer to expire when the request may be granted without exceeding the schedulable entity's maximum allotment. Upon timer expiry, the rate controller module allows the request to be passed on to the scheduler, where it will be scheduled for service.

The implementation of the rate controller module requires a separate queue and timer for each schedulable entity, requiring a large amount of memory for storing the state of each timer and checking each timer to see if it has expired. The state space required scales linearly with the number of schedulable entities, and can become burdensome for large numbers of schedulable entities. As the number of resource users, each corresponding to a different schedulable entity, increases, the state space required to manage the scheduler and allocate resources properly increases rapidly.

20

Thus, it is desirable to provide a system and method for resource scheduling capable of limiting schedulable entities to a certain minimum and maximum resource allocation, without

requiring the significant state space required by a typical rate controller module. The system should also ensure that resources are shared fairly among the different schedulable entities.

00/0000" 9/999999

SUMMARY OF THE INVENTION

The present invention schedules resource requests from a plurality of schedulable entities while limiting the maximum and minimum quality of service allocated to each schedulable entity. One embodiment of a scheduler in accordance with the present invention requires less
5 memory to maintain state information than existing rate-controlling schedulers, and is thus more easily scalable to large numbers of users. The scheduler also schedules resources fairly among competing schedulable entities.

A resource scheduler uses a fair-share scheduling algorithm to select resource requests to service from multiple request queues, each associated with a schedulable entity. After a resource request is selected from a queue, a rate controller checks to ensure that servicing the request will not cause the associated user's maximum quality of service to be exceeded. If the maximum quality of service will not be exceeded, the request is serviced and the virtual time is incremented. If the maximum quality of service will be exceeded, the virtual time is still incremented, but the actual request is not serviced and remains pending.

The features and advantages described in the specification are not all-inclusive, and particularly, many additional features and advantages will be apparent to one of ordinary skill in the art in view of the drawings, specification, and claims hereof. Moreover, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive
20 subject matter, resort to the claims being necessary to determine such inventive subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is an illustration of a resource request scheduler adapted to limit the maximum quality of service allocated to each schedulable entity.

Fig. 2 is a flowchart of the process for selecting and servicing requests from schedulable
5 entities while limiting the maximum quality of service allocated to each schedulable entity.

Fig. 3 is an illustration of a hierarchical resource request scheduler adapted to limit the maximum quality of service allocated to each schedulable entity.

The figures depict a preferred embodiment of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention described herein.

2020032510360

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Reference will now be made in detail to several embodiments of the present invention, examples of which are illustrated in the accompanying drawings. Wherever practicable, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

5 Before describing various embodiments of the present invention, some further background on scheduling methods is provided.

Any separately identifiable source of requests for a resource is referred to herein as a "schedulable entity." As examples, a schedulable entity may represent a single individual, a group of individuals with some shared association, a computer or a set of computer programs associated with a resource. For example, a company A may contain two divisions Y and Z. If resource requests, such as requests for network bandwidth, are only identified as originating from company A, company A is a "schedulable entity," and all resource requests from both division Y and division Z are included in the same scheduling queue. However, if company A network bandwidth requests may be separately identified and scheduled between division Y and division Z, then both Y and Z are schedulable entities. (Separate qualities of service may be assigned to the different divisions, and requests from each division are placed into separate scheduling queues.

15 Different quality of service guarantees may be assigned to different types of resources for the same schedulable entity. Many different types of resources may be assigned quality of service guarantees, such as CPU time, memory access time, file access time, and networking resources. Other types of resources will be evident to one of skill in the art. For example, schedulable entity "Entity A" may have a CPU quality of service set to 50% of the physical

computer's resources. Entity A may also be assigned only 40% of the physical resource's memory access time. In another embodiment, a schedulable entity may have a single quality of service guarantee that applies to each type of resource. For example, Entity A may be assigned a minimum quality of service of 40%, and a maximum quality of service of 50% for all resources.

5 In this case, Entity A will receive between 40 and 50% of all of the resources of the physical computer.

The quality of service assigned to each schedulable entity for each resource type is stored in a quality of service table or similar data structure. The quality of service is expressed as either a minimum and maximum percentage share of resources, or as a minimum and maximum quantity of a particular resource. In one embodiment, the minimum and maximum quality of service are equal, and a single quality of service value bounds the resources allocated to a particular schedulable entity. The present invention does not limit how quality of service parameters are set or selected for entities, and so any mechanism for managing quality of service may be used.

Fair-share scheduling algorithms are well known in the art, and numerous different variations exist. Fair-share scheduling algorithms partition a resource among multiple users such that each user is allocated a fair share of the resource. For purposes of example, the start-time fair queuing algorithm with virtual time scheduling will be discussed herein. However, it will be understood by one of skill in the art that numerous other fair-share scheduling algorithms may be

20 used with the inventive techniques disclosed herein. For example, a round-robin, a deficit round-robin, or a self-clocked fair queuing algorithm may be used.

Certain fair-share scheduling algorithms use various methods to emulate the idealized scheduling discipline of generalized processor sharing (GPS). In GPS, each schedulable entity

has a separate queue. The scheduler serves each non-empty queue by servicing an infinitesimally small amount of data from each queue in turn. Each queue may be associated with a service weight, and the queue receives service in proportion to this weight. Weighted fair queuing algorithms simulate the properties of GPS by calculating the time at which a request would receive service under GPS, and then servicing requests in order of these calculated service times. The start-time fair queuing algorithm is a variation of weighted fair queuing.

A virtual time may be used in fair-share scheduling algorithms. A virtual time scheduler increments a virtual time variable. A start and finish tag is calculated for each incoming resource request, and requests are serviced in order of their tags. The virtual time is equal to the current request's start tag number, and increments as additional start tags are calculated. A virtual time scheduler simulates time-division multiplexing of requests in much the same way as weighted fair queuing algorithms simulate the properties of GPS. Virtual time scheduling, weighted fair queuing, and the start-time fair queuing algorithm are discussed in "An Engineering Approach to Computer Networking" by S. Keshav, pp. 209-263, (Addison-Wesley Professional Computing Series) (1997), and "Start-time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks" by Pawan Goyal, Harrick M. Vin, and Haichen Cheng, IEEE/ACM Transactions on Networking, Vol. 5, No. 5, pp. 690-704 (October 1997), the subject matter of both of which are hereby incorporated in their entirety.

The start-time fair queuing algorithm is used with virtual time scheduling in the following manner. Incoming resource requests are placed in separate queues within a resource scheduler, with each queue holding the requests from a particular schedulable entity. Each resource request is tagged with both a start number (SN) and a finish number (FN) tag as it reaches the head of its respective queue, and requests are serviced in order of increasing start

numbers. The start and finish number tags are calculated for each request k from a schedulable entity i , where i represents the queue in which the request k has been placed. The start and finish number tags are calculated using the virtual time $V(t)$, the weight given to each schedulable entity $\Phi(i)$ (if weighting is being implemented) and the resource duration $D(i, k, t)$ for which each request is scheduled.

The virtual time $V(t)$ is initially zero. When the resource scheduler is busy, the virtual time at time t is defined to be equal to the start tag of the request in service at time t . At the end of a busy period, the virtual time is set to the maximum finish tag assigned to any resource request that has been serviced.

The start number tag SN of a request k arriving at an inactive queue i (one that does not currently contain previous pending requests by the queue's schedulable entity) is set to the maximum of either the current virtual time $V(t)$ or the finish number FN of the previous request $(k-1)$ pending in the queue:

$$SN(i, k, t) = \max[V(t), FN(i, k-1, t)] \quad (1)$$

The finish number tag FN of a request k is the sum of the start number SN of the request and its request duration $D(i, k, t)$ divided by its weight $\Phi(i)$:

$$FN(i, k, t) = SN(i, k, t) + \frac{D(i, k, t)}{\Phi(i)} \quad (2)$$

The weight $\Phi(i)$ for each schedulable entity corresponds to the minimum quality of service assigned to that schedulable entity. Schedulable entities are provided with resources in proportion to their weights, and thus a schedulable entity with a minimum quality of service guarantee of 40% of a resource receives proportionally more resources than another schedulable

entity with a minimum quality of service guarantee of 20% of a resource. As shown in Eq. 2, as $\Phi(i)$ increases towards 1 (100% of the resources available), the amount $D(i,k,t)/\Phi(i)$ added to the SN decreases, and thus the finish number FN tag is lower for the current request ($k-1$). As shown in Eq. 1, a lower finish number FN tag on the current request ($k-1$) causes the next request k to have a lower start number SN tag. A lower SN tag means that the next request k is serviced more quickly. Thus increasing the weight of a particular queue increases the frequency at which requests are serviced from the queue.

In one embodiment, each schedulable entity queue i has an individually assigned minimum quality of service, and thus each schedulable entity is allotted a different proportion of the overall resources. In another embodiment, all of the schedulable entities are allocated the same minimum quality of service guarantee, and weights are not implemented.

The request duration D for which a resource request is scheduled is specific to the type of resource being requested and the particular scheduling system implementation. Request duration D determines the granularity of resource scheduling. For example, assume a process P requires a duration of 100 seconds of CPU time. Scheduling process P continuously for 100 seconds would mean that all other processes would starve for 100 seconds, an undesirable result. Instead, process P will typically be scheduled for a shorter upper bound duration D_{max} , such as 20 seconds. After 20 seconds, process P is preempted and another process is scheduled. If the original duration D requested is shorter than the upper bound duration D_{max} , the entire requested duration D will be serviced. Additionally, the process may block on I/O earlier than the upper bound duration on CPU time.

Various embodiments of the present invention will next be discussed. Fig. 1 illustrates a scheduler suitable for scheduling the shared usage of a variety of different types of

resources. For example, scheduler 100 may be used to schedule resources for CPU time, memory access, disk access, or networking resources such as bus bandwidth. Additional types of resources suitable for scheduling with the scheduler 100 will be evident to one of skill in the art. Different schedulable entities make requests for resources, such as a request for CPU time, a request for memory access, a request for disk access, or a request for bandwidth to transport signals within the network.

Scheduler 100 selects resource requests for service using a fair-share scheduling algorithm. Scheduler 100 additionally prevents a selected resource request from being serviced if the maximum quality of service assigned to the schedulable entity that made the resource request would thereby be exceeded. Scheduler 100 implements rate control over the maximum quality of service in a manner that preserves the fairness properties of the scheduling algorithm. Scheduler 100 further implements rate control in a manner that reduces the memory required to store state variables as compared to prior-art methods. Scheduler 100 may be implemented as part of the operating system of a computer.

Different quality of service guarantees are implemented by allocating different amounts of the scheduled resource to servicing each of the schedulable entities. Resources may be allocated to different schedulable entities as a percentage of a particular resource, for example, allocating 50% of the resource to schedulable entity A and 25% to schedulable entity B. Resources may also be allocated as a particular number of units of a resource, for example, the operating system may be instructed to allocate x seconds of memory access to schedulable entity A and y seconds of memory access to schedulable entity B.

Each schedulable entity has an assigned minimum and maximum quality of service guarantee for the resource being scheduled. The minimum guarantee represents the minimum amount of a particular resource the schedulable entity should receive. The maximum guarantee represents the maximum amount of a particular resource the schedulable entity should receive, and this maximum will be enforced even if the resource will become idle. In one embodiment, the minimum and maximum quality of service guarantees are equal. In another embodiment, the maximum quality of service exceeds the minimum quality of service, for example, schedulable entity A is guaranteed at least 20% of the resource, but at no time will entity A receive more than 30% of the resource.

Scheduler 100 receives incoming resource requests 110 from a group of schedulable entities who share the resource being scheduled. The incoming requests 110 are sorted into separate queues, with each queue holding the pending requests for a single schedulable entity. Three queues 130, 140 and 150 are shown in Fig. 1. Queue 130 holds two pending resource requests 132A and 132B. Queue 140 holds three pending resource requests 142A, 142B, and 142C. Queue 150 holds one pending resource request 152A. It will be evident to one of skill in the art that additional queues may be added to the scheduler 100 if additional schedulable entities are to share the resource being scheduled.

Requests 132A, 142A and 152A reside in the head of queues 130, 140, and 150, respectively. As resource requests are serviced, they leave the head of the queue, and requests remaining in the queue move up in the queue. The fair-share selector 180 selects resource requests for service from the head of each queue based upon a fair-share scheduling algorithm, which ensures that each schedulable entity receives its minimum quality of service. Each resource request is assigned a start number tag *SN* and finish number tag *FN* (Eqs. 1 and 2) as it

reaches the head of its respective queue. Selector 180 selects the next resource request to service as the one with the lowest SN tag. The fair-share selector 180 will not allocate service time to an empty queue.

If the schedulable entities have equal minimum quality of service guarantees (equal weights), the fair-share scheduling algorithm apportions resources equally among the schedulable entities. If the minimum quality of service guarantees for the schedulable entities differ, each schedulable entity has a weight $\Phi(i)$ that is incorporated into the fair-share scheduling algorithm, as shown in Eq. 2. The weight $\Phi(i)$ thus influences the tags assigned to each resource request and the resulting selection order of requests.

In the following example, it will be assumed that the scheduler 100 uses the start-time fair queuing algorithm with a virtual clock tracking the virtual time $V(t)$. Selector 180 also limits each selected request to a pre-determined maximum duration D_{max} , thereby limiting the amount of resource time allocated to any single request.

As resource requests enter the head of each queue, the scheduler 100 assigns each request a start number tag SN using Eq. 1, and a finish number tag FN using Eq. 2. Selector 180 selects requests for service based upon their start number SN order, with the lowest SN selected first. Ties are broken arbitrarily. Each request includes a request duration $D_{request}$. The request will be scheduled for service for a duration $D=D_{request}$; however, if $D_{request}$ is greater than the scheduler 100's pre-determined upper bound duration D_{max} , the selector 180 will only permit D_{max} of the request to be selected for service:

$$D = \min(D_{request}, D_{max}) \quad (3)$$

If $D_{request} > D_{max}$, the remainder of the request ($D_{remainder} = D_{request} - D_{max}$) will be returned to the head of its queue, and a new start number and finish number tag will be calculated for the remainder of the request.

The virtual time $V(t)$ is related to the current request's start number SN . Each time that the selector 180 selects 114 a request for service with a new start number tag SN , this advances the virtual time $V(t)$ of the scheduler 100. As shown in Eq. 1, this calculation of SN depends on the finish number FN calculation given in Eq. 2. A component of the FN calculation is the request duration $D(i,k,t)$, and thus the request duration also influences the virtual time $V(t)$.

Once a request has been selected 114 for service, it is checked 116 by the rate controller for its respective queue. Queue 130 has a rate controller 136, queue 140 has a rate controller 146, and queue 150 has a rate controller 156. Each rate controller checks to determine whether the selected request is eligible for service, i.e. whether servicing the selected request will exceed the maximum quality of service guarantee for the associated queue's schedulable entity. Techniques for determining whether satisfying the current selected request would result in the request's schedulable entity exceeding its pre-specified maximum quality of service are well known in the art. Rate controlled schedulers are discussed in "An Engineering Approach to Computer Networking" by S. Keshav, pp. 248-252, (Addison-Wesley Professional Computing Series) (1997), the subject matter of which is herein incorporated by reference in its entirety.

If servicing the request would exceed the maximum quality of service guarantee, the request is not eligible for service and the rate controller leaves the request pending at the head of its respective queue. However, the rate controller will send a dummy request to be serviced 120 that is scheduled for a zero time duration D , thereby updating the virtual time. The SN and FN tags for the request left pending at the head of the non-eligible queue will thus be recalculated as

if the request had just arrived at the head of the queue. This SN and FN tag recalculation occurs both for requests that are not eligible for service, and for request remainders as discussed previously.

If the rate controller determines that the request is eligible for service, the request is removed from its queue. The request is then serviced, meaning that the request is allowed to consume the resource being scheduled for a duration D .

Scheduler 100 employs rate controllers 136, 146 and 156 to ensure that each schedulable entity does not exceed its maximum quality of service guarantee. The rate controllers do not require a separate set of rate controller queues, nor does each rate controller require a separate timer. Thus, the rate controllers require less memory for state variable storage as compared to prior-art rate control mechanisms.

Fig. 2 is a flowchart of the process for selecting and servicing resource requests as implemented within a resource scheduling module, such as the scheduler 100. The method of Fig. 2 will be illustrated by an example of scheduling a CPU resource using scheduler 100 of Fig. 1. For purposes of example, assume that the CPU resource's D_{max} is 20 seconds. The SN tag, and $D_{request}$ associated with each head-of-queue resource request is given in Table 1 below:

<u>SN tag</u>	<u>Resource request</u>	<u>$D_{request}$ (seconds)</u>
1	132A	50
3	142A	20
6	152A	30

Table 1

The scheduler reviews 200 the *SN* tags for the head-of-queue requests (132A, 142A, and 152A). The scheduler selects 210 the resource request with the smallest or “next” *SN* tag (132A). In one embodiment, ties are broken arbitrarily. In another embodiment, the system administrator specifies an order for resolving tag number ties. The scheduler then determines 215 the duration D to allot to the selected resource request. In this example, since $D_{request} > D_{max}$ (50 seconds > 20 seconds), the request 132A will only be granted D_{max} (20 seconds) of CPU time.

The rate controller (136) associated with the queue of the selected request (132A) is called 220. The rate controller determines 230 whether servicing the selected request will exceed the maximum quality of service guarantee allocated to the request’s schedulable entity. If the maximum quality of service will be exceeded, the scheduler sends a dummy request 250 for servicing, thereby simulating servicing the request within the fair-share scheduling algorithm controlling the request selection process. The virtual time $V(t)$ then advances to the *SN* tag of the next request, just as if the request 132A had actually been serviced. If the maximum quality of service will not be exceeded, the resource request is actually serviced 240.

5 The scheduler then calculates 260 new tags as needed. If the entire requested duration $D_{request}$ of the selected request (132A) was serviced 240, resource request 132B moves to the head of queue 130 and a new SN and FN tag is assigned to request 132B. However, if the entire request or part of the request (132A) was not serviced and is still pending, new tags are calculated for the portion of request 132A that was not serviced. In this example, 30 seconds of original request 132A remain to be serviced. The remainder of request 132A with an updated duration $D_{request}$ of 30 seconds remains pending in the queue 130. New SN and FN tags are calculated for the remainder of request 132A, and request 132B remains back in the queue 130.

The scheduler then returns to step 200 and reviews the head-of-queue SN tags to select the next request for service.

10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995
1000
1005
1010
1015
1020
1025
1030
1035
1040
1045
1050
1055
1060
1065
1070
1075
1080
1085
1090
1095
1100
1105
1110
1115
1120
1125
1130
1135
1140
1145
1150
1155
1160
1165
1170
1175
1180
1185
1190
1195
1200
1205
1210
1215
1220
1225
1230
1235
1240
1245
1250
1255
1260
1265
1270
1275
1280
1285
1290
1295
1300
1305
1310
1315
1320
1325
1330
1335
1340
1345
1350
1355
1360
1365
1370
1375
1380
1385
1390
1395
1400
1405
1410
1415
1420
1425
1430
1435
1440
1445
1450
1455
1460
1465
1470
1475
1480
1485
1490
1495
1500
1505
1510
1515
1520
1525
1530
1535
1540
1545
1550
1555
1560
1565
1570
1575
1580
1585
1590
1595
1600
1605
1610
1615
1620
1625
1630
1635
1640
1645
1650
1655
1660
1665
1670
1675
1680
1685
1690
1695
1700
1705
1710
1715
1720
1725
1730
1735
1740
1745
1750
1755
1760
1765
1770
1775
1780
1785
1790
1795
1800
1805
1810
1815
1820
1825
1830
1835
1840
1845
1850
1855
1860
1865
1870
1875
1880
1885
1890
1895
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050
2055
2060
2065
2070
2075
2080
2085
2090
2095
2100
2105
2110
2115
2120
2125
2130
2135
2140
2145
2150
2155
2160
2165
2170
2175
2180
2185
2190
2195
2200
2205
2210
2215
2220
2225
2230
2235
2240
2245
2250
2255
2260
2265
2270
2275
2280
2285
2290
2295
2300
2305
2310
2315
2320
2325
2330
2335
2340
2345
2350
2355
2360
2365
2370
2375
2380
2385
2390
2395
2400
2405
2410
2415
2420
2425
2430
2435
2440
2445
2450
2455
2460
2465
2470
2475
2480
2485
2490
2495
2500
2505
2510
2515
2520
2525
2530
2535
2540
2545
2550
2555
2560
2565
2570
2575
2580
2585
2590
2595
2600
2605
2610
2615
2620
2625
2630
2635
2640
2645
2650
2655
2660
2665
2670
2675
2680
2685
2690
2695
2700
2705
2710
2715
2720
2725
2730
2735
2740
2745
2750
2755
2760
2765
2770
2775
2780
2785
2790
2795
2800
2805
2810
2815
2820
2825
2830
2835
2840
2845
2850
2855
2860
2865
2870
2875
2880
2885
2890
2895
2900
2905
2910
2915
2920
2925
2930
2935
2940
2945
2950
2955
2960
2965
2970
2975
2980
2985
2990
2995
3000
3005
3010
3015
3020
3025
3030
3035
3040
3045
3050
3055
3060
3065
3070
3075
3080
3085
3090
3095
3100
3105
3110
3115
3120
3125
3130
3135
3140
3145
3150
3155
3160
3165
3170
3175
3180
3185
3190
3195
3200
3205
3210
3215
3220
3225
3230
3235
3240
3245
3250
3255
3260
3265
3270
3275
3280
3285
3290
3295
3300
3305
3310
3315
3320
3325
3330
3335
3340
3345
3350
3355
3360
3365
3370
3375
3380
3385
3390
3395
3400
3405
3410
3415
3420
3425
3430
3435
3440
3445
3450
3455
3460
3465
3470
3475
3480
3485
3490
3495
3500
3505
3510
3515
3520
3525
3530
3535
3540
3545
3550
3555
3560
3565
3570
3575
3580
3585
3590
3595
3600
3605
3610
3615
3620
3625
3630
3635
3640
3645
3650
3655
3660
3665
3670
3675
3680
3685
3690
3695
3700
3705
3710
3715
3720
3725
3730
3735
3740
3745
3750
3755
3760
3765
3770
3775
3780
3785
3790
3795
3800
3805
3810
3815
3820
3825
3830
3835
3840
3845
3850
3855
3860
3865
3870
3875
3880
3885
3890
3895
3900
3905
3910
3915
3920
3925
3930
3935
3940
3945
3950
3955
3960
3965
3970
3975
3980
3985
3990
3995
4000
4005
4010
4015
4020
4025
4030
4035
4040
4045
4050
4055
4060
4065
4070
4075
4080
4085
4090
4095
4100
4105
4110
4115
4120
4125
4130
4135
4140
4145
4150
4155
4160
4165
4170
4175
4180
4185
4190
4195
4200
4205
4210
4215
4220
4225
4230
4235
4240
4245
4250
4255
4260
4265
4270
4275
4280
4285
4290
4295
4300
4305
4310
4315
4320
4325
4330
4335
4340
4345
4350
4355
4360
4365
4370
4375
4380
4385
4390
4395
4400
4405
4410
4415
4420
4425
4430
4435
4440
4445
4450
4455
4460
4465
4470
4475
4480
4485
4490
4495
4500
4505
4510
4515
4520
4525
4530
4535
4540
4545
4550
4555
4560
4565
4570
4575
4580
4585
4590
4595
4600
4605
4610
4615
4620
4625
4630
4635
4640
4645
4650
4655
4660
4665
4670
4675
4680
4685
4690
4695
4700
4705
4710
4715
4720
4725
4730
4735
4740
4745
4750
4755
4760
4765
4770
4775
4780
4785
4790
4795
4800
4805
4810
4815
4820
4825
4830
4835
4840
4845
4850
4855
4860
4865
4870
4875
4880
4885
4890
4895
4900
4905
4910
4915
4920
4925
4930
4935
4940
4945
4950
4955
4960
4965
4970
4975
4980
4985
4990
4995
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
5085
5090
5095
5100
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
5155
5160
5165
5170
5175
5180
5185
5190
5195
5200
5205
5210
5215
5220
5225
5230
5235
5240
5245
5250
5255
5260
5265
5270
5275
5280
5285
5290
5295
5300
5305
5310
5315
5320
5325
5330
5335
5340
5345
5350
5355
5360
5365
5370
5375
5380
5385
5390
5395
5400
5405
5410
5415
5420
5425
5430
5435
5440
5445
5450
5455
5460
5465
5470
5475
5480
5485
5490
5495
5500
5505
5510
5515
5520
5525
5530
5535
5540
5545
5550
5555
5560
5565
5570
5575
5580
5585
5590
5595
5600
5605
5610
5615
5620
5625
5630
5635
5640
5645
5650
5655
5660
5665
5670
5675
5680
5685
5690
5695
5700
5705
5710
5715
5720
5725
5730
5735
5740
5745
5750
5755
5760
5765
5770
5775
5780
5785
5790
5795
5800
5805
5810
5815
5820
5825
5830
5835
5840
5845
5850
5855
5860
5865
5870
5875
5880
5885
5890
5895
5900
5905
5910
5915
5920
5925
5930
5935
5940
5945
5950
5955
5960
5965
5970
5975
5980
5985
5990
5995
6000
6005
6010
6015
6020
6025
6030
6035
6040
6045
6050
6055
6060
6065
6070
6075
6080
6085
6090
6095
6100
6105
6110
6115
6120
6125
6130
6135
6140
6145
6150
6155
6160
6165
6170
6175
6180
6185
6190
6195
6200
6205
6210
6215
6220
6225
6230
6235
6240
6245
6250
6255
6260
6265
6270
6275
6280
6285
6290
6295
6300
6305
6310
6315
6320
6325
6330
6335
6340
6345
6350
6355
6360
6365
6370
6375
6380
6385
6390
6395
6400
6405
6410
6415
6420
6425
6430
6435
6440
6445
6450
6455
6460
6465
6470
6475
6480
6485
6490
6495
6500
6505
6510
6515
6520
6525
6530
6535
6540
6545
6550
6555
6560
6565
6570
6575
6580
6585
6590
6595
6600
6605
6610
6615
6620
6625
6630
6635
6640
6645
6650
6655
6660
6665
6670
6675
6680
6685
6690
6695
6700
6705
6710
6715
6720
6725
6730
6735
6740
6745
6750
6755
6760
6765
6770
6775
6780
6785
6790
6795
6800
6805
6810
6815
6820
6825
6830
6835
6840
6845
6850
6855
6860
6865
6870
6875
6880
6885
6890
6895
6900
6905
6910
6915
6920
6925
6930
6935
6940
6945
6950
6955
6960
6965
6970
6975
6980
6985
6990
6995
7000
7005
7010
7015
7020
7025
7030
7035
7040
7045
7050
7055
7060
7065
7070
7075
7080
7085
7090
7095
7100
7105
7110
7115
7120
7125
7130
7135
7140
7145
7150
7155
7160
7165
7170
7175
7180
7185
7190
7195
7200
7205
7210
7215
7220
7225
7230
7235
7240
7245
7250
7255
7260
7265
7270
7275
7280
7285
7290
7295
7300
7305
7310
7315
7320
7325
7330
7335
7340
7345
7350
7355
7360
7365
7370
7375
7380
7385
7390
7395
7400
7405
7410
7415
7420
7425
7430
7435
7440
7445
7450
7455
7460
7465
7470
7475
7480
7485
7490
7495
7500
7505
7510
7515
7520
7525
7530
7535
7540
7545
7550
7555
7560
7565
7570
7575
7580
7585
7590
7595
7600
7605
7610
7615
7620
7625
7630
7635
7640
7645
7650
7655
7660
7665
7670
7675
7680
7685
7690
7695
7700
7705
7710
7715
7720
7725
7730
7735
7740
7745
7750
7755
7760
7765
7770
7775
7780
7785
7790
7795
7800
7805
7810
7815
7820
7825
7830
7835
7840
7845
7850
7855
7860
7865
7870
7875
7880
7885
7890
7895
7900
7905
7910
7915
7920
7925
7930
7935
7940
7945
7950
7955
7960
7965
7970
7975
7980
7985
7990
7995
8000
8005
8010
8015
8020
8025
8030
8035
8040
8045
8050
8055
8060
8065
8070
8075
8080
8085
8090
8095
8100
8105
8110
8115
8120
8125
8130
8135
8140
8145
8150
8155
8160
8165
8170
8175
8180
8185
8190
8195
8200
8205
8210
8215
8220
8225
8230
8235
8240
8245
8250
8255
8260
8265
8270
8275
8280
8285
8290
8295
8300
8305
8310
8315
8320
8325
8330
8335
8340
8345
8350
8355
8360
8365
8370
8375
8380
8385
8390
8395
8400
8405
8410
8415
8420
8425
8430
8435
8440
8445
8450
8455
8460
8465
8470
8475
8480
8485
8490
8495
8500
8505
8510
8515
8520
8525
8530
8535
8540
8545
8550
8555
8560
8565
8570
8575
8580
8585
8590
8595
8600
8605
8610
8615
8620
8625
8630
8635
8640
8645
8650
8655
8660
8665
8670
8675
8680
8685
8690
8695
8700
8705
8710
8715
8720
8725
8730
8735
8740
8745
8750
8755
8760
8765
8770
8775
8780
8785
8790
8795
8800
8805
8810
8815
8820
8825
8830
8835
8840
8845
8850
8855
8860
8865
8870
8875
8880
8885
8890
8895
8900
8905
8910
8915
8920
8925
8930
8935
8940
8945
8950
8955
8960
8965
8970
8975
8980
8985
8990
8995
9000
9005
9010
9015
9020
9025
9030
9035
9040
9045
9050
9055
9060
9065
9070
9075
9080
9085
9090
9095
9100
9105
9110
9115
9120
9125
9130
9135
9140
9145
9150
9155
9160
9165
9170
9175
9180
9185
9190
9195
9200
9205
9210
9215
9220
9225
9230
9235
9240
9245
9250
9255
9260
9265
9270
9275
9280
9285
9290
9295
9300
9305
9310
9315
9320
9325
9330
9335
9340
9345
9350
9355
9360
9365
9370
9375
9380
9385
9390
9395
9400
9405
9410
9415
9420
9425
9430
9435
9440
9445
9450
9455
9460
9465
9470
9475
9480
9485
9490
9495
9500
9505
9510
9515
9520
9525
9530
9535
9540
9545
9550
9555
9560
9565
9570
9575
9580
9585
9590
9595
9600
9605
9610
9615
9620
9625
9630
9635
9640
9645
9650
9655
9660
9665
9670
9675
9680
9685
9690
9695
9700
9705
9710
9715
9720
9725
9730
9735
9740
9745
9750
9755
9760
9765
9770
9775
9780
9785
9790
9795
9800
9805
9810
9815
9820
9825
9830
9835
9840
9845
9850
9855
9860
9865
9870
9875
9880
9885
9890
9895
9900
9905
9910
9915
9920
9925
9930
9935
9940
9945
9950
9955
9960
9965
9970
9975
9980
9985
9990
9995
10000
10005
10010
10015
10020
10025
10030
10035
10040
10045
10050
10055
10060
10065
10070
10075
10080
10085
10090
10095
10100
10105
10110
10115
10120
10125
10130
10135
10140
10145
10150
10155
10160
10165
10170
10175

assume Company A requests that a minimum of 40% of the resources of a particular CPU be guaranteed for the Company A, and Company A is prohibited from using more than 45% of the resources. Company A also wishes to ensure that its Division X receives at least 60% of the Company A CPU resources, and that its Division Y receives the remainder of the resources.

5 The hierarchical resource scheduler will constrain Company A to using between 40 and 45% of the available resources of the shared CPU, using a weighted fair-share queuing algorithm and rate controllers as described in Figs. 1 and 2 to implement the desired minimum and maximum quality of service. Company A, however, may request that the resource scheduler implement one of several different methods for resource sharing between Divisions X and Y. In one embodiment, Divisions X and Y are assigned resources using a non-work-conserving scheduling algorithm wherein both X and Y are constrained to a minimum and a maximum quality of service. In another embodiment, Divisions X and Y are assigned resources using a work-conserving scheduling algorithm wherein both X and Y are guaranteed a minimum quality of service, and any additional resources are shared between X and Y according to their respective weights.

Fig. 3 illustrates a hierarchical fair-share scheduler 300. A parent scheduler 302 has two child schedulers 301A and 301B. Parent scheduler 302 includes two parent queues 380A and 380B, corresponding to two schedulable entities. Child scheduler 301A includes two child queues 330A and 330B, corresponding to two schedulable entities, both of which feed resource requests 320A to parent queue 380A. Child scheduler 301B includes three child queues 340A, 340B and 340C, all of which feed resource requests 320B to parent queue 380B. Each parent queue represents a main schedulable entity (such as a company), and each child queue represents a subgroup schedulable entity of its parent (such as a division of the company).

Initial resource requests 310 are separated by child schedulable entity and placed into their corresponding child queues 330 or 340. Child schedulers 301A and 301B assign start and finish number tags to requests in the heads of their respective queues. Each child scheduler 301 maintains a separate set of tags, and the parent scheduler 302 also maintains a separate set of tags. Consequently, each scheduler has a separate virtual time clock. Selector 326A selects resource requests for service from requests at the head of queues 330A and 330B using a fair-share scheduling algorithm. Selector 326B also selects resource requests for service from requests at the head of queues 340A, 340B and 340C using a fair-share scheduling algorithm.

In one embodiment, each child queue is assigned a weight $\Phi(i)$ increasing or decreasing the queue's relative resource share. Each child queue also has an associated rate controller that limits the maximum resource share that that child queue may obtain. Child queue 330A has a rate controller 336A; child queue 330B has a rate controller 336B; child queue 340A has a rate controller 346A; child queue 340B has a rate controller 346B; and child queue 340C has a rate controller 346C. When a request from the head of a child queue is selected for service, the associated rate controller determines if servicing the request will exceed the maximum quality of service allocated to the child queue. If the maximum quality of service will be exceeded, a dummy request for zero resources is sent for service as a placeholder, and the request remains pending in the head of its child queue. Start and finish number tags are updated as described previously, thereby incrementing the virtual time for the scheduler.

Requests (including dummy requests) selected for service by selector 326A that are not blocked by their respective rate controllers are output 320A into queue 380A of scheduler 302. Similarly, requests selected for service by selector 326B that are not blocked by their respective rate controllers are output 320B into queue 380B of scheduler 302. Scheduler 302 assigns new

start and finish number tags to requests in queues 380A and 380B, thereby incrementing the virtual time for the parent scheduler 302. A selector 328 uses a fair-share scheduling algorithm to select requests for service from the heads of queues 380A and 380B. Queue 380A has an associated rate controller 386A, and queue 380B has an associated rate controller 386B.

5 Scheduler 302 uses the method described in Fig. 2 to output resource requests for servicing 322, subject to minimum and maximum quality of service constraints on queues 380A and 380B.

In the embodiment shown in Fig. 3, child schedulers 301A and 301B are implemented in a manner similar to the parent scheduler 302. In another embodiment, the hierarchical resource scheduler 300 implements one or more of the child schedulers 301A and 301B differently than parent scheduler 302. For example, child schedulers 301A and/or 301B may be implemented without rate controllers associated with each child queue. A child queue without a rate controller will not be subject to a maximum quality of service limitation. Additionally, child schedulers 301A and/or 301B may be implemented as work-conserving schedulers, or may use different types of scheduling algorithms.

The resource scheduling method of the present invention is suitable for use in scheduling the resources of "virtual servers." It is desirable for an ISP to provide multiple server applications on a single physical host computer, in order to allow multiple customers to use a single host computer. If a different customer is associated with each server application, or "virtual server", the ISP will implement a method of sharing resources between customers.

20 Additionally, it is desirable to be able to constrain virtual server customers to a minimum and maximum quality of service guarantee. This allows customers to be limited to a certain amount of the resources of the physical host computer.

The resource scheduler of the present invention may be used in the context of virtual servers to schedule some or all of the physical host computer resources. Each virtual server corresponds to a separate schedulable entity. Each virtual server is assigned a maximum and minimum quality of service guarantee. In another embodiment, separate maximum and minimum quality of service guarantees may be assigned to different resources used by the same virtual server. Requests for resources from each virtual server are serviced according to the resource scheduling method of the present invention.

Although the invention has been described in considerable detail with reference to certain embodiments, other embodiments are possible. As will be understood by those of skill in the art, the invention may be embodied in other specific forms without departing from the essential characteristics thereof. For example, different fair-share algorithms may be implemented in the resource request scheduler. Additionally, a weighted fair-share or hierarchical weighted fair-share algorithm implementation may be used in the resource request scheduler. Accordingly, the present invention is intended to embrace all such alternatives, modifications and variations as fall within the spirit and scope of the appended claims and equivalents.